

A SPECTRAL ASSIGNMENT APPROACH FOR THE GRAPH ISOMORPHISM PROBLEM

STEFAN KLUS AND TUHIN SAHAI*

Abstract. In this paper, we propose a heuristic for the graph isomorphism problem that is based on the eigendecomposition of the adjacency matrices. It is well known, that the eigenvalues of the adjacency matrices of isomorphic graphs need to be identical. However, two graphs \mathcal{G}_A and \mathcal{G}_B can be isospectral but non-isomorphic. If the graphs possess repeated eigenvalues, which typically correspond to graph symmetries, finding isomorphisms is much harder. By repeatedly perturbing the adjacency matrices, it is possible to break symmetries of the graphs without changing the isomorphism and to assign vertices of \mathcal{G}_A to vertices of \mathcal{G}_B , provided that an admissible assignment exists. This heuristic approach can be used to construct a permutation which transforms \mathcal{G}_A into \mathcal{G}_B if the graphs are isomorphic, or to show that no isomorphism exists.

Key words. Graph isomorphism problem, highly regular graphs, linear assignment problem

AMS subject classifications. 05C60, 05C50

1. Introduction. Two graphs $\mathcal{G}_A = (\mathcal{V}_A, \mathcal{E}_A)$ and $\mathcal{G}_B = (\mathcal{V}_B, \mathcal{E}_B)$ with adjacency matrices A and B are defined to be isomorphic if there exists a permutation $\pi \in \mathcal{S}_n$ such that

$$(1.1) \quad (v_i, v_j) \in \mathcal{E}_A \Leftrightarrow (v_{\pi(i)}, v_{\pi(j)}) \in \mathcal{E}_B,$$

or, equivalently, if the adjacency matrices are permutation-similar, i.e.

$$(1.2) \quad B = P^T A P,$$

with $P \in \mathcal{P}_n$. Here, n is the number of vertices, \mathcal{S}_n the symmetric group of degree n , and \mathcal{P}_n the set of all $n \times n$ permutation matrices. The relation between the permutation π and the permutation matrix $P = (p_{ij})$ is given by

$$(1.3) \quad p_{ij} = \begin{cases} 1, & \text{if } \pi(i) = j, \\ 0, & \text{otherwise.} \end{cases}$$

If the graphs \mathcal{G}_A and \mathcal{G}_B are isomorphic – denoted by $\mathcal{G}_A \cong \mathcal{G}_B$ –, then the number of isomorphisms from \mathcal{G}_A to \mathcal{G}_B is identical to the number of automorphisms of \mathcal{G}_A or \mathcal{G}_B , respectively [1]. There exists no polynomial-time algorithm to check whether two arbitrary graphs are isomorphic. Interestingly, the graph isomorphism (GI) problem is one of only a few problems for which the complexity class is unknown [2]. The complexity of the best known exact method for isomorphism testing is $2^{\sqrt{O(n \log n)}}$, as described in [3]. In the past, different authors have tried to develop polynomial-time algorithms and to prove that GI belongs to the complexity class P . The validity of these methods seems to be yet unknown. It has also not been shown that the graph isomorphism problem is NP -complete. GI belongs to the larger family of isomorphism problems on algebraic structures such as groups or rings that seem to lie between P and NP -complete [4]. Another open question is whether GI can be solved efficiently using quantum computers. The graph isomorphism problem can be regarded as a

*United Technologies Research Center, 411 Silver Lane, East Hartford, CT 06108, USA (kluss@utrc.utc.com, sahait@utrc.utc.com)

non-Abelian hidden subgroup problem (HSP) where the hidden subgroup is the automorphism group of the graph. An efficient solution of the HSP, which is the basis of many quantum algorithms, is only known for certain Abelian groups, whereas the general non-Abelian case remains open [5]. For several special classes, however, the graph isomorphism problem is known to be solvable in polynomial time. These classes include, for instance, planar graphs [6] and graphs with bounded degree [7] or bounded eigenvalue multiplicity [8]. For an overview of isomorphism testing methods for these restricted graph classes, we refer to [2].

In this paper, we develop an eigendecomposition-based approach for the graph isomorphism problem and show how local perturbations of the adjacency matrices can be used to break symmetries and to identify possible assignments. To this end, we construct a matrix C which contains the costs of assigning vertex v_i of \mathcal{G}_A to vertex v_j of \mathcal{G}_B and solve a linear assignment problem (LAP). Our algorithm extends the applicability of other spectral methods for the graph isomorphism problem to strongly regular graphs with repeated eigenvalues. We will assume that the eigenvalues and eigenvectors of the adjacency matrices can be computed with a sufficiently high accuracy.

The paper is organized as follows: Section 2 contains a brief overview of spectral properties of graphs and how these properties can be used to construct a heuristic for graph isomorphism testing. In Section 3, we propose a novel eigendecomposition-based algorithm to determine whether two graphs are isomorphic. Numerical results for a number of different graphs including strongly regular graphs and isospectral but non-isomorphic graphs are presented in Section 4. Section 5 lists open questions and possible future work.

2. Eigenvalues and eigenvectors of graphs. The graph isomorphism problem can be written as a combinatorial optimization problem of the form

$$(2.1) \quad \min_{P \in \mathcal{P}_n} \|A - P^T B P\|_F,$$

where $\|\cdot\|_F$ denotes the Frobenius norm¹. The graphs \mathcal{G}_A and \mathcal{G}_B are isomorphic if and only if the minimum of the above cost function is zero.

Case 1: Graphs with distinct eigenvalues. Let $\mathcal{O}_n = \{P \in \mathbb{R}^{n \times n} \mid P^T P = I\} \subset \mathcal{P}_n$ denote the set of all orthogonal matrices. If the matrices A and B are symmetric – we consider only undirected graphs –, then the solution of the relaxed problem

$$(2.2) \quad \min_{P \in \mathcal{O}_n} \|A - P^T B P\|_F,$$

which is called two-sided orthogonal Procrustes problem [9], can be computed analytically, provided that A and B have distinct eigenvalues. This result is captured in the following theorem:

THEOREM 2.1. *Given two symmetric matrices A and B whose eigenvalues are distinct, let $A = V_A \Lambda_A V_A^T$ and $B = V_B \Lambda_B V_B^T$ be the eigendecompositions, with $\Lambda_A = \text{diag}(\lambda_A^{(1)}, \dots, \lambda_A^{(n)})$, $\Lambda_B = \text{diag}(\lambda_B^{(1)}, \dots, \lambda_B^{(n)})$, and $\lambda_A^{(1)} < \dots < \lambda_A^{(n)}$ as well as $\lambda_B^{(1)} < \dots < \lambda_B^{(n)}$. Then the orthogonal matrix P^* which minimizes (2.2) is given by*

$$(2.3) \quad P^* = V_B S V_A^T,$$

¹The Frobenius norm of a matrix A is defined as $\|A\|_F = \sqrt{\text{tr}(A^T A)}$.

where

$$(2.4) \quad S = \text{diag}(\pm 1, \dots, \pm 1).$$

A proof of this theorem can be found in [9], for example. Note that the eigenvalues and corresponding eigenvectors have to be sorted both in either increasing or decreasing order. Now, let c be the cost of the optimal solution of the relaxed problem, i.e.

$$(2.5) \quad c = \min_{P \in \mathcal{O}_n} \|A - P^T B P\|_F = \|A - P^{*T} B P^*\|_F = \|\Lambda_A - \Lambda_B\|_F.$$

There exist 2^n different solutions with the same cost. If the graphs \mathcal{G}_A and \mathcal{G}_B are isomorphic, then one of these 2^n solutions is the permutation matrix we are looking for. As $\mathcal{P}_n \subset \mathcal{O}_n$, the graphs cannot be isomorphic if $c \neq 0$. If, on the other hand, $c = 0$, this implies that the eigenvalues of A and B are identical and the graphs \mathcal{G}_A and \mathcal{G}_B are isospectral, but not necessarily isomorphic. An example of isospectral graphs, taken from [10], is shown in Figure 1.

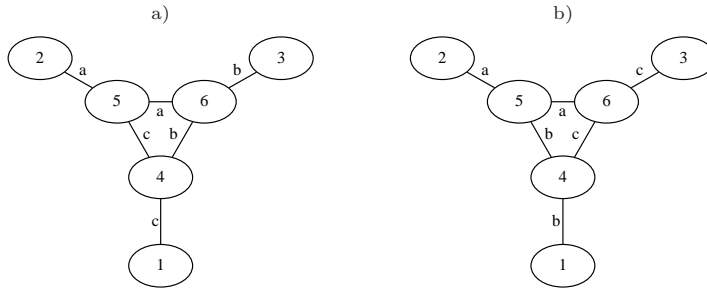


FIG. 1. *Isospectral graphs.*

In addition to the eigenvalues, the eigenvectors of the graphs can be used for isomorphism testing [1]. Setting $a = 1$, $b = 2$, and $c = 3$, the eigenvectors of the graphs belonging to the largest eigenvalue $\lambda_A^{(n)} = \lambda_B^{(n)} = 5.167$ are

$$\begin{aligned} v_A^{(n)} &= [0.380, 0.092, 0.157, 0.655, 0.477, 0.407]^T, \\ v_B^{(n)} &= [0.222, 0.068, 0.352, 0.575, 0.353, 0.606]^T. \end{aligned}$$

Since the entries of the eigenvectors are different, $v_B^{(n)}$ cannot be written as a permutation of $v_A^{(n)}$, implying that \mathcal{G}_A and \mathcal{G}_B are not isomorphic. Provided that the entries of the eigenvectors are distinct, this simple test can be used for graph isomorphism testing. Here, we compared the entries of the eigenvectors $v_A^{(n)}$ and $v_B^{(n)}$ since the largest eigenvalue of the adjacency matrix of a connected graph always has multiplicity one [11]. Furthermore, the corresponding eigenvector is strictly positive. Note that in general the normalized eigenvectors are only determined up to the sign. Thus, two comparisons might be required. If the absolute values of all entries of an eigenvector are different – Spielman [1] calls such an eigenvector *helpful* –, this gives us a canonical labeling of the vertices and the graph isomorphism problem can be solved easily.

An isomorphism testing algorithm for graphs with n distinct eigenvalues developed by Leighton and Miller [12], which was never published, is presented in [1]. The method determines the diagonal entries ± 1 of the matrix S such that $P^* = V_B S V_A^T$ is a permutation matrix by breaking vertices that are not equivalent into different classes. Based on the entries of the eigenvectors, these classes are refined until an isomorphism is found or the graphs are shown to be non-isomorphic.

Case 2: Graphs with repeated eigenvalues. If the graphs possess repeated eigenvalues, finding isomorphisms is much harder. The eigenvectors belonging to repeated eigenvalues are unique only up to basis rotations. Repeated eigenvalues typically correspond to graph symmetries, although the correspondence is not exact [11]. A graph is said to be *regular* if each vertex has the same number of neighbors and *strongly regular* if additionally integers a and b exist such that every pair of vertices v_i and v_j shares exactly a common neighbors if the vertices v_i and v_j are adjacent and exactly b common neighbors otherwise. It can be shown that these strongly regular graphs possess at most three distinct eigenvalues, see for example [11].

Let $\lambda_A = [\lambda_A^{(1)}, \dots, \lambda_A^{(m)}]$ be the eigenvalues of the graph \mathcal{G}_A with multiplicities $\mu_A = [\mu_A^{(1)}, \dots, \mu_A^{(m)}]$, i.e. $\sum_{k=1}^m \mu_A^{(k)} = n$, and let V_A be partitioned into

$$(2.6) \quad V_A = [V_A^{(1)}, \dots, V_A^{(m)}].$$

That is, $V_A^{(k)} \in \mathbb{R}^{n \times \mu_A^{(k)}}$ is either the eigenvector belonging to the eigenvalue $\lambda_A^{(k)}$ or the matrix whose columns form an orthogonal basis of the eigenspace. The eigenvalues of the Paley graph shown in Figure 2, for example, are $\lambda_A = [\frac{-1-\sqrt{17}}{2}, \frac{-1+\sqrt{17}}{2}, 8]$ with multiplicities $\mu_A = [8, 8, 1]$. Finding isomorphisms for this graph proved to be particularly difficult due to the symmetries and highly regular structure. For graphs with several symmetries or repeated eigenvalues, comparing only eigenvectors belonging to distinct eigenvalues is not sufficient. Therefore, we will also exploit additional information encoded in matrices representing orthogonal projections onto the eigenspace of repeated eigenvalues.

Let $\lambda_A^{(k)}$ be a repeated eigenvalue of graph \mathcal{G}_A . For two vertices v_i and v_j , define $V_A^{(k)}(v_i)$ and $V_A^{(k)}(v_j)$ to be the rows i and j of the matrix $V_A^{(k)}$. These row vectors clearly depend on the orthogonal basis chosen for the eigenspace, but the scalar product is independent of the choice of basis [13]. The convex hull of all vectors $V_A^{(k)}(v_i)$, $i = 1, \dots, n$, is called the eigenpolytope of the graph belonging to the eigenvalue $\lambda_A^{(k)}$. The matrix

$$(2.7) \quad E_A^{(k)} = V_A^{(k)} (V_A^{(k)})^T,$$

i.e. $(E_A^{(k)})_{ij} = \langle V_A^{(k)}(v_i), V_A^{(k)}(v_j) \rangle$, represents the orthogonal projection onto the column space of $V_A^{(k)}$ and is an invariant of the eigenspace [14]. It holds that $A E_A^{(k)} = E_A^{(k)} A = \lambda_A^{(k)} E_A^{(k)}$. Furthermore, the matrix A can be written as

$$(2.8) \quad A = \sum_{k=1}^m \lambda_A^{(k)} E_A^{(k)}.$$

For a detailed description of the relation between a graph and the geometry of its eigenpolytopes, we refer to [13, 14]. We now want to exploit properties of the matrices $E_A^{(k)}$ to identify isomorphisms. In the next section we will show that by comparing eigenvectors and eigenpolytopes, it is possible to compute isomorphisms of strongly regular graphs such as the Paley graph shown in Figure 2.

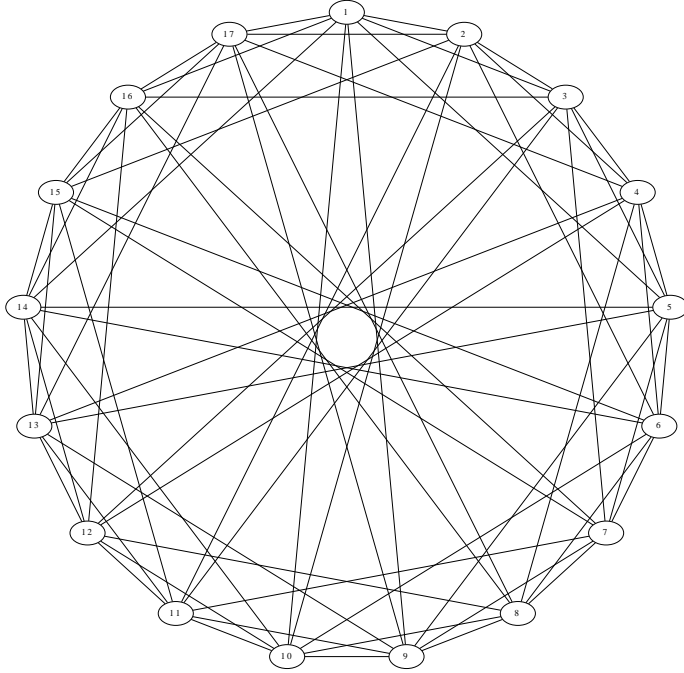


FIG. 2. *Strongly regular Paley graph.*

3. Spectral approach for the graph isomorphism problem. As described in the previous section, repeated eigenvalues complicate graph isomorphism testing. Our approach is based on finding local perturbations of the adjacency matrices A and B that break symmetries without changing the isomorphism. Let us illustrate the basic idea with a simple example. Given the two cycle graphs shown in Figure 3a, we would like to find an isomorphism from \mathcal{G}_A to \mathcal{G}_B . The eigenvalues of the cycle graph are $\lambda_A = [-2, -1, 1, 2]$ with multiplicities $\mu_A = [1, 2, 2, 1]$. Here, it is not possible to use the eigenvectors directly to determine the permutation since the entries of the normalized distinct eigenvectors are all $\pm 1/\sqrt{6}$. In order to find an assignment for vertices of graph \mathcal{G}_A to vertices of \mathcal{G}_B , we perturb the adjacency matrices A and B by adding self-loops with different weights. If we add a self-loop to vertex v_1 of \mathcal{G}_A and a self-loop to vertex v_1 of graph \mathcal{G}_B , the two graphs remain isomorphic. Now all eigenvalues are distinct and the eigenvectors of A can still be written as a – non-unique – permutation of the eigenvectors of B . Thus, we assign vertex v_1 of \mathcal{G}_A to vertex v_1 of \mathcal{G}_B . Note that due to the cyclic symmetry, we could also assign vertex v_1 of \mathcal{G}_A to any other vertex of \mathcal{G}_B . The updated graphs are shown in Figure 3b, the red vertices denote self-loops with weight 1.

Now, we use the new adjacency matrices and try to assign vertex v_2 of \mathcal{G}_A to a vertex of \mathcal{G}_B . Since we have broken the cyclic symmetry, there are now only two possible assignments, vertex v_2 of \mathcal{G}_A could be either assigned to vertex v_5 or v_6 of \mathcal{G}_B . If we try to assign it to a different vertex, the eigenvalues of the adjacency matrices differ. Adding self-loops with weight 2 to vertex v_2 of \mathcal{G}_A and vertex v_5 of \mathcal{G}_B – shown in Figure 3c –, the entries of the eigenvectors are distinct and the permutation could be extracted directly. Alternatively, the procedure described above can be repeated until an assignment for all vertices is found. For the remaining vertices, the assignment is

unambiguous and the resulting graphs are shown in Figure 3d.

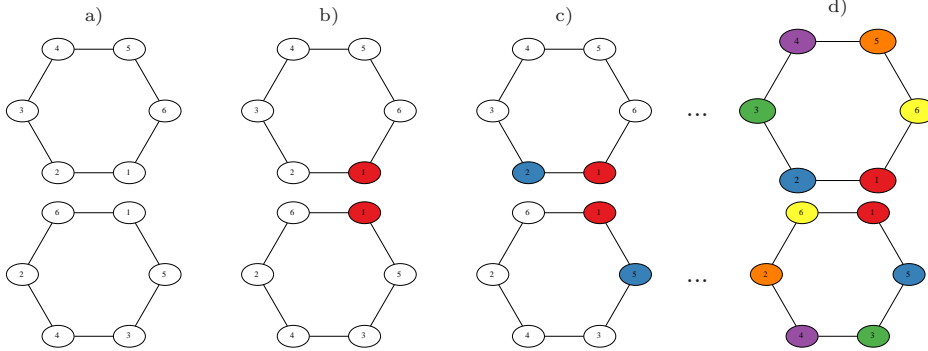


FIG. 3. Graph isomorphism testing procedure. The various colors represent self-loops with different weights.

Let us formalize the above procedure. In order to perturb the adjacency matrices A and B and hence the eigenvalues and eigenvectors, we use single-entry matrices representing self-loops with different weights w .

DEFINITION 3.1. Define $D_i(w) = \text{diag}(d_1, \dots, d_n)$ to be the diagonal matrix with $d_j = w \delta_{ij}$, i.e.

$$(3.1) \quad d_j = \begin{cases} w, & \text{if } j = i, \\ 0, & \text{otherwise.} \end{cases}$$

We start with the original adjacency matrices A and B and construct cost matrices $C^{(k)}$ as follows. One possibility to define the entries $c_{ij}^{(k)}$ for distinct eigenvalues $\lambda_A^{(k)}$ would be to make the signs of the corresponding eigenvectors consistent and to compare the eigenvectors entry-wise using

$$(3.2) \quad c_{ij}^{(k)} = \left| V_A^{(k)}(v_i) - V_B^{(k)}(v_j) \right|.$$

To avoid problems with the determination of consistent signs of eigenvectors, for example if $V_A^{(k)}$ and $-V_A^{(k)}$ contain exactly the same entries, we use a slightly different approach to compute the cost $c_{ij}^{(k)}$. Instead of comparing the eigenvectors directly, we compute the projection matrices $E_A^{(k)}$ and $E_B^{(k)}$ and check for each row i of $E_A^{(k)}$ whether it can be written as a permutation of row j of $E_B^{(k)}$ by comparing the sorted vectors.

DEFINITION 3.2. Let $E_A^{(k)}(v_i)$ and $E_B^{(k)}(v_j)$ be the i -th and j -th row of the matrices $E_A^{(k)}$ and $E_B^{(k)}$, respectively, and let $s : \mathbb{R}^n \rightarrow \mathbb{R}^n$ be a function that sorts the entries of a vector in ascending order. Then $c_{ij}^{(k)}$ is defined as

$$(3.3) \quad c_{ij}^{(k)} = \left\| s \left(E_A^{(k)}(v_i) \right) - s \left(E_B^{(k)}(v_j) \right) \right\|_F.$$

The same definition is also used for repeated eigenvalues. We then compute the sum of all these cost matrices, i.e.

$$(3.4) \quad C = \sum_{k=1}^m C^{(k)},$$

where the entry c_{ij} of C now describes the cost of assigning v_i of \mathcal{G}_A to v_j of \mathcal{G}_B . We have observed that the number of zero entries of the matrix C is related to the number of isomorphisms between the two graphs. To compute possible assignments, we solve the resulting linear assignment problem

$$(3.5) \quad \min_{P \in \mathcal{P}_n} \text{tr}(C^T P).$$

For the cycle graph shown in Figure 3, the structure of the matrices C that result in successful assignments are shown in Figure 4. Without a perturbation of the adjacency matrices, each vertex of \mathcal{G}_A can be assigned to each vertex of \mathcal{G}_B and the cost matrix C is zero. After one iteration, there exist only two possible assignments, and after two iterations, the solution is unique.

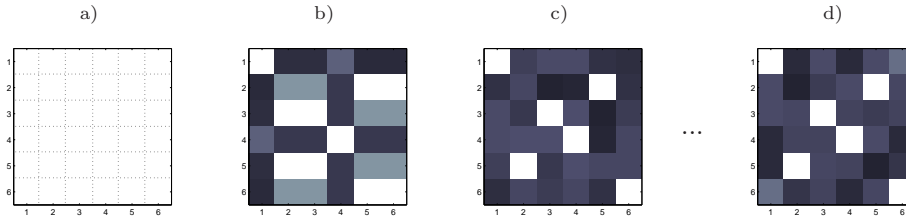


FIG. 4. Structure of the cost matrices C for the cycle graph. White entries correspond to possible assignments with cost $c_{ij} < \varepsilon$.

If the cost of the solution of the LAP is zero, this means that a possible assignment exists. We then use the new perturbed adjacency matrices and try to find assignments for the remaining vertices. The solution of the LAP itself, however, is not necessarily a valid solution of the graph isomorphism problem since the vertices cannot be assigned independently. Consider, for example, Figure 4b. If we assign vertex v_2 to vertex v_5 , then the assignment of v_3 to v_2 becomes invalid. If we, on the other hand, assign vertex v_2 to vertex v_6 , then the assignment of v_3 to v_3 becomes invalid. Therefore, we assign the vertices iteratively. If the graphs are isomorphic and the algorithm found valid assignments for all vertices, then the diagonals of the adjacency matrices A and B determine the permutation π .

A detailed description of the proposed graph isomorphism testing approach is presented in Algorithm 1 and Algorithm 2. In the description of the algorithm, we have not included backtracking techniques. Backtracking is needed if a previous assignment was successful but does not result in a correct permutation. We then delete the previous assignment and try to find a different assignment for this vertex. The algorithm is stopped if no assignment for vertex v_1 of \mathcal{G}_A could be found.

Without backtracking, the overall runtime of the algorithm is $O(n^6 \log n)$. The eigenvalues and eigenvectors of the adjacency matrices can be computed in $O(n^3)$, sorting the eigenvectors is $O(n \log n)$. The linear assignment problem can also be solved in $O(n^3)$ using the Hungarian method [15, 16]. Our proposed algorithm requires at most $2n^2$ eigendecompositions and n^2 solutions of a LAP. The number of eigendecompositions and solutions of LAPs could be reduced to $n(n+1)$ and $\frac{1}{2}n(n+1)$, respectively, by considering only vertices that have not been assigned during previous iterations. Furthermore, the procedure can be stopped if the solution of the LAP is unique. The number of iterations required to obtain a unique solution depends on the order in which the vertices are perturbed.

Algorithm 1 Graph isomorphism testing.

```

function ISISOMORPHIC( $A, B$ )
   $e_0 := \text{FINDPERMUTATION}(A, B)$                                 // see Algorithm 2
  if  $e_0 > \varepsilon$  then
    return false                                              // initial graphs differ
  end if
  for  $i = 1, \dots, n$  do
     $\tilde{A} := A + D_i(i)$ 
    for  $j = 1, \dots, n$  do
       $\tilde{B} := B + D_j(i)$ 
       $e_{ij} := \text{FINDPERMUTATION}(\tilde{A}, \tilde{B})$                     // see Algorithm 2
      if  $e_{ij} < \varepsilon$  then
         $A := \tilde{A}$                                               // assign vertex  $i$  of  $\mathcal{G}_A$  ...
         $B := \tilde{B}$                                               // ... to vertex  $j$  of  $\mathcal{G}_B$ 
        break
      end if
    end for
    if  $e_{ij} > \varepsilon \forall j = 1, \dots, n$  then
      return false                                          // no possible assignment found
    end if
  end for
  return true                                              // assignment for all vertices found
end function

```

4. Results. In this section, we will present numerical results for benchmark problems downloaded from [17] and [18]. For our computations, we used *Matlab* and an error tolerance $\varepsilon = 10^{-6}$. That is, two eigenvalues of a matrix are defined to be identical if the difference is less than ε . Furthermore, an assignment is accepted if the cost e of the solution of the LAP is less than ε .

In order to illustrate the proposed graph isomorphism testing approach, we use the Paley graph shown in Figure 2 and a given permutation of the graph. Without a perturbation of the adjacency matrices, all entries of the cost matrix C are zero. The structure of the cost matrices after 1, 2, 3, and 4 assignments is shown in Figure 4. Here, the resulting permutation π is given by

$$\pi = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\ 1 & 6 & 15 & 3 & 11 & 7 & 17 & 12 & 9 & 8 & 4 & 2 & 5 & 16 & 10 & 13 & 14 \end{bmatrix}.$$

This shows that the method successfully computes isomorphisms of strongly regular graphs with repeated eigenvalues. For all graphs downloaded from [17], the algorithm returned correct results without backtracking.

Results for larger benchmark problems used by *conauto* are presented in Table 1. For each benchmark problem, we run the proposed algorithm 100 times using different randomly generated permutations of the original graph. Here, n is the number of vertices, nBT the number of runs where no backtracking was required, and BT the number of runs where backtracking was required to find an isomorphism. The column *steps* describes the average number of backtracking steps needed to find an isomorphism and the last column the average runtime. The efficiency of the algorithm could be easily improved using C or C++ and the techniques proposed at the end of Section 3. The results show that the algorithm returns correct results for most of the benchmark problems without backtracking. Only Steiner triple system graphs and unions of strongly regular graphs required backtracking for almost all test cases.

Algorithm 2 Check whether permutations exist.

```

function FINDPERMUTATION( $A, B$ )
  Decompose  $A = V_A \Lambda_A V_A^T$  and  $B = V_B \Lambda_B V_B^T$ 
   $e := \|\Lambda_A - \Lambda_B\|_F$ 
  if  $e > \varepsilon$  then
    return  $e$                                      // eigenvalues not identical
  end if
   $C := 0 \in \mathbb{R}^{n \times n}$ 
  for  $k = 1, \dots, m$  do
     $E_A^{(k)} := V_A^{(k)} (V_A^{(k)})^T$  and  $E_B^{(k)} := V_B^{(k)} (V_B^{(k)})^T$ 
    for  $i = 1, \dots, n$  do
       $u_A := i$ -th row of  $E_A^{(k)}$ 
      for  $j = 1, \dots, n$  do
         $u_B := j$ -th row of  $E_B^{(k)}$ 
         $e := \text{CHECKPERMUTATION}(u_A, u_B)$ 
         $c_{ij} := c_{ij} + e$ 
      end for
    end for
  end for
   $e := \min_{P \in \mathcal{P}_n} \text{tr}(C^T P)$                                      // solve LAP
  return  $e$ 
end function

function CHECKPERMUTATION( $u_A, u_B$ )
   $\tilde{u}_A := s(u_A)$  and  $\tilde{u}_B := s(u_B)$ 
   $e := \|\tilde{u}_A - \tilde{u}_B\|_F$ 
  return  $e$ 
end function

```

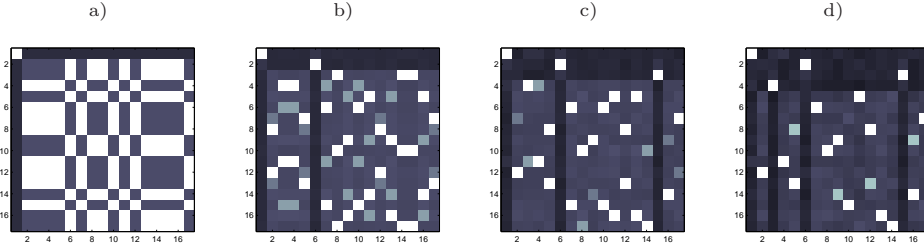


FIG. 5. Structure of the cost matrices C for the Paley graph after 1, 2, 3, and 4 successful assignments. White entries represent assignments with $c_{ij} < \varepsilon$.

5. Conclusion. In this work, we have presented an eigendecomposition-based heuristic for solving the graph isomorphism problem that relies on the repeated perturbation of the adjacency matrices and the solution of linear assignment problems. The capability of the algorithm was demonstrated with the aid of several strongly regular graphs and further benchmark problems. We believe that the proposed approach can be used to find isomorphisms efficiently, to detect and break symmetries, and to gain insight into the structure of highly regular graphs.

Future work includes a theoretical analysis of the algorithm that investigates for which classes of graphs the proposed algorithm returns correct results without the use of backtracking techniques. Backtracking could result in an exponential worst-case

TABLE 1
Test results for various benchmark graphs.

Type	Name	n	nBT	BT	steps (avg)	time
Steiner triple system graphs	sts-19_57	57	5	95	37.91	66.580011
Latin square graphs (prime order)	latin-3_9	9	100	0	0.00	0.019826
	latin-5_25	25	100	0	0.00	0.313762
	latin-7_49	49	98	2	1.00	3.386035
Latin square graphs (prime power order)	latin-2_4	4	100	0	0.00	0.003229
	latin-4_16	16	100	0	0.00	0.081307
	latin-6_36	36	74	26	2.31	1.213612
Paley graphs (prime order)	paley-prime_13	13	89	11	1.09	0.046544
	paley-prime_29	29	100	0	0.00	0.512827
Paley graphs (prime power order)	paley-power_9	9	100	0	0.00	0.018404
	paley-power_25	25	100	0	0.00	0.316294
Lattice graphs	lattice(4)_16	16	100	0	0.00	0.078996
	lattice(6)_36	36	100	0	0.00	1.036952
Triangular graphs	triangular(7)_21	21	100	0	0.00	0.178016
	triangular(10)_45	45	100	0	0.00	2.313508
Unions of strongly regular graphs	usr(1)_29-1	29	11	89	13.49	3.644380
Clique-connected cubic	chh_cc(1-1)_22-1	22	100	0	0.00	0.148856
hypo-Hamiltonian graphs	chh_cc(2-1)_44-1	44	100	0	0.00	1.926242
Non-disjoint unions of undirected tripartite graphs	tnn(1)_26-1	26	100	0	0.00	0.316860
	tnn(2)_52-1	52	100	0	0.00	5.007201
Random graphs	iso_r01N_s20	20	100	0	0.00	0.079963
	iso_r01N_s40	40	100	0	0.00	0.850544

runtime. Another open question is whether it is possible to compute the eigenvalues and eigenvectors of large graphs with a sufficiently high accuracy.

REFERENCES

- [1] D. A. Spielman. Spectral graph theory (Lecture notes). <http://www.cs.yale.edu/homes/spielman>, 2009.
- [2] J. Köbler. On graph isomorphism for restricted graph classes. In A. Beckmann, U. Berger, B. Löwe, and J. Tucker, editors, *Logical Approaches to Computational Barriers*, Lecture Notes in Computer Science, pages 241–256. Springer, 2006.
- [3] D. A. Spielman. Faster isomorphism testing of strongly regular graphs. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 576–584. ACM, 1996.
- [4] V. Arvind and J. Torán. Isomorphism testing: Perspective and open problems. *Bulletin of the European Association for Theoretical Computer Science*, 86:66–84, 2005.
- [5] S. Hallgren, A. Russell, and A. Ta-Shma. The hidden subgroup problem and quantum computation using group representations. *SIAM Journal on Computing*, 32(4):916–934, 2003.
- [6] J. E. Hopcroft and R. E. Tarjan. Isomorphism of planar graphs. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 131–152. Plenum Press, 1972.
- [7] E. M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25(1):42–65, 1982.
- [8] L. Babai, D. Y. Grigoryev, and D. M. Mount. Isomorphism of graphs with bounded eigenvalue multiplicity. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, pages 310–324. ACM, 1982.
- [9] P. Schönemann. On two-sided orthogonal Procrustes problems. *Psychometrika*, 33(1):19–33, 1968.
- [10] I. Oren and R. Band. Isospectral graphs with identical nodal counts. *Journal of Physics A: Mathematical and Theoretical*, 45(13):1–11, 2012.
- [11] L. Lovász. Eigenvalues of graphs. Technical report, Eötvös Loránd University, Budapest, Hungary, 2007.
- [12] F. T. Leighton and G. L. Miller. Certificates for graphs with distinct eigenvalues. Original Manuscript, 1979.
- [13] C. D. Godsil. Eigenpolytopes of distance regular graphs. *Canadian Journal of Mathematics*, 50(4):739–755, 1998.
- [14] A. Chan and C. D. Godsil. Symmetry and eigenvectors. In G. Hahn and G. Sabidussi, editors, *Graph Symmetry: Algebraic Methods and Applications*. Kluwer, 1997.

- [15] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.
- [16] R. E. Burkard and E. Çela. Linear assignment problems and extensions. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*. Kluwer, 1999.
- [17] V. V. Valiayeu. *Griso* for regular graphs. <http://sourceforge.net/projects/griso>.
- [18] J. L. López-Presa, A. F. Anta, and L. N. Chiroque. Graph isomorphism algorithm *conauto*. <https://sites.google.com/site/giconauto>.